Sym Train Simulation Intelligence

Automated Customer Assistance Pipeline

December 2025 | Final Project Submission

Keywords: Large Language Models, ETL, Few-Shot Learning

1. Project Overview & Objectives

The Problem Statement

Customer service agents frequently encounter complex inquiries that require immediate, accurate procedural guidance. Manual search for resolution steps is inefficient.

Proposed Solution

Development of an automated pipeline utilizing **Generative AI** to extract insights from historical transcripts and generate context-aware resolution steps for new inquiries.

2. System Architecture & Tech Stack

Data Engineering

Python 3.10+
Pandas (ETL)
JSON
Parsing

Model Layer

DeepSeek (LLM)
HuggingFace
Transformers
Few-Shot Logic

Deployment

Streamlit UI
Docker
Containers
GitHub Actions

3. Data Ingestion Methodology

Source Data Structure:

The dataset consists of hierarchical JSON files. Key extraction targets include audioContentItems, sequence numbers, and speaker roles.

Transformation Logic:

- Recursive parsing of nested JSON structures.
- Concatenation of multi-turn dialogues into single "Full Transcript" strings.
- Normalization of speaker labels (Trainee vs. Sym).

4. Information Extraction: Comparative Analysis

We evaluated two distinct approaches for extracting "Reason" and "Resolution Steps" from the transcripts.

Methodology	Model Used	Observations
LLM-Based	DeepSeek (OpenAI API)	High fidelity; adhered strictly to JSON schema via system prompts. captured nuance well.
Transformer Baseline	Google Flan-T5	Limited by 2500 char context window. Output required extensive regex post-processing.

5. Intent Classification

Target Classes

- Update Payment Method
- File Insurance Claim
- Track Order Status
- Other

Algorithm Selection

1. Rule-Based: Boolean keyword matching. High precision, low recall.

2. Zero-Shot (BART): facebook/bart-large-mnli. Computationally expensive but generalizes well.

3. LLM: Semantic understanding provided the highest accuracy for ambiguous queries.

6. Few-Shot Generation Architecture

The core "intelligence" engine follows a Retrieve-Then-Generate workflow.

1. Input

New User Query

2. Classify

Predict Intent

3. Retrieve

Fetch *k=3*

Examples

4. Prompt

Inject Context

5. Output

JSON Response

7. Implementation Challenges & Solutions

Strict JSON Enforcement

LLMs tend to include conversational filler ("Here is the JSON..."). We utilized response_format={"type": "json_object"} combined with robust error handling to ensure parseable outputs.

API Latency & Limits

Batch processing of large transcript files triggered rate limits (HTTP 429). We implemented exponential backoff strategies and inserted time.sleep() intervals between API calls.

Conclusion & Future Work

Summary

The project successfully demonstrates that Few-Shot Learning significantly outperforms Zero-Shot baselines for complex procedural generation in customer support contexts.

Future Directions

1. **Vector Databases:** Replace category-based retrieval with semantic embedding search (RAG).
2. **Multimodal Integration:** Map generated text steps to specific UI screenshots (Vision).